# Hierarchical multi-classification

Hendrik Blockeel[1], Maurice Bruynooghe[1], Sašo Džeroski[2], Jan Ramon[1], and Jan Struyf[1]

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A, B-3001 Leuven, Belgium
{Hendrik.Blockeel,Maurice.Bruynooghe,Jan.Ramon,Jan.Struyf}@cs.kuleuven.ac.be
Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si

**Abstract.** The problem of hierarchical multi-classification is considered. In this setting a set of classes is to be assigned to a single instance, and all possible classes are structured according to a hierarchy. Example application domains are functional genomics and text classification. An algorithm is presented to solve hierarchical multi-classification tasks. It is a decision tree induction algorithm that is based on the notion of predictive clustering trees and in which a suitable distance measure is plugged in. Preliminary results with the algorithm on data sets from functional genomics and text classification are reported and discussed.

## 1 Introduction

Many data mining algorithms construct predictive models where the target variable is one-dimensional and either nominal (for classification tasks) or numerical (for regression tasks). Few algorithms can handle the task of making predictions of a more structural kind. Even relational data mining algorithms usually focus on learning classification or regression models where structural properties are used to make the prediction, but the predicted value itself is not structured.

Multi-classification is a simple kind of structure prediction, where the target variable is a set of classes. The setting occurs relatively frequently, e.g. in document classification (a document is typically relevant for several topics, hence it may belong to several classes). The problem can be tackled by learning separate models for each class (indicating whether a single instance belongs to a class or not), but learning a single model for all classes has the advantage that the total size of the predictive theory is typically smaller, and dependencies between different classes w.r.t. membership can be taken into account and may even be explicitated. Advantages of learning a single model for multiple related prediction tasks have been reported several times in the literature (see e.g. [5] for decision trees, [7, 1] for neural networks, [24] for text classification).

While our setting is one of assigning a set of classes to an instance, we look in more detail at the case where the classes are ordered in a hierarchy. This

hierarchy concisely conveys relevant information about the similarity and differences between classes and also expresses the constraint that an object belonging to a class also belongs to the parent class. Using a hierarchy, a prediction can be represented as a subtree of the hierarchy with the predicted classes as the leaves (and internal nodes). This boils down to a special case of prediction of structured values, an extension of the multi-classification problem, that we call *hierarchical multi-classification*. For example, this is useful in document classification when a document is to be classified into a number of newsgroups: newsgroups indeed form a hierarchy.

The remainder of this text is structured as follows. In Section 2 we describe the general setting of distance-based multi-classification, of which hierarchical multi-classification (Section 3) is a special case. Section 4 briefly reviews the ideas behind predictive clustering trees, on which our algorithm will be based. Section 5 presents a distance measure that is to be plugged into the decision tree learner in order to obtain multi-classification. Section 6 presents experiments with this algorithm. Section 7 discusses related work and Section 8 concludes.

## 2 Distance-based Multi-classification

Our approach to hierarchical multi-classification is placed in the more general context of distance-based multi-classification, which we describe first.

In the multi-classification context each individual belongs to one or more classes, and given a new instance we wish to predict the classes to which it belongs. An important element here is how to evaluate the quality of predictions. We distinguish several possibilities, from more general to more specific:

- Cost-based. A cost can be associated with (a) including class $i$ in the prediction when it should not be there ($CI_i$); (b) omitting class $i$ when it should be included ($CO_i$); (c) substituting class $i$ for class $j$ ($CS_{ij}$). Costs are positive.
- Distance-based. This is similar to the cost-based approach but imposes the constraint that $CO_i = CI_i$ and $CS_{ij} = CS_{ji}$ for all classes $i$ and $j$ (due to the symmetry of distances.) Such distances are similar to so-called edit distances.
- Accuracy-based. No substitutions are allowed, they are modelled using inclusions and omissions ($CI_i = CO_j$ and $CS_{ij}$ is undefined). This criterion amounts to a kind of weighted average accuracy with which the different classes are predicted.

*Example 1.* Consider the task of predicting for a given article for which newsgroups it is relevant. Assume the relevant newsgroups for some article are {`rec.autos`, `science.electronics`}. How good are the predictions {`rec.autos, science`}, {`rec, science.electronics`} and {`rec.sport, science.electronics`} respectively? Is the cost of making a more vague prediction (`rec` instead of `rec.autos`) higher than that of making a more specific but wrong guess (`rec.sport`)? Is a mistake in classification on high level worse than one on a lower level? Costs will have to be chosen based on the answers to these questions, which may depend on the application.

*Example 2.* Now consider the task of composing an ice-cream cone for someone. We have to choose a set of flavours that the person will like. There might be a penalty for omitting chocolate, which the person likes. Assume the person likes strawberry, and would normally not choose raspberry. Given the similarity between raspberry and strawberry, the penalty for substituting raspberry for strawberry is probably smaller than the sum of the penalties for including raspberry and omitting strawberry. This situation cannot be modelled using the accuracy-based method.

The current paper studies hierarchical multi-classification in which the distances between classes are defined by their position in a hierarchy. This is an instance of the "distance-based" setting mentioned above.

## 3 Hierarchical Multi-classification

We represent a hierarchy on a set of values $C$ as a tree, defined by a root element and a *parent* function that maps children of a tree node onto the node (it is not defined for the root element). A *valid set* is a set of values that is closed with respect to the *parent* function, i.e., if $c \in S$ then $parent(c) \in S$ or $c$ is the root. $2^C$ denotes the powerset of $C$ and $V(C) \subseteq 2^C$ denotes the set of valid sets in $C$. The problem of hierarchical multi-classification can then be stated as follows.
**Given:**

- an instance space $X$
- a class space $C$
- a hierarchy on $C$
- a set of examples $D \subseteq X \times V(C)$
- a quality criterion $Q$

**Find:** a function $h : X \to V(C)$ that maps an instance $x$ onto a valid set of classes $S$, so that $h$ maximizes the quality criterion $Q$.

If we represent the hierarchy as a tree, then a single prediction is a subtree (in the graph sense) of it. Figure 1 illustrates this. In the case of predicting a single leaf class in the hierarchy, the subtree reduces to a single branch.

We assume every subtree in a hierarchy is a semantically correct and meaningful prediction. For specific applications, some constraints may hold, for example that some classes are incompatible. If so that information can better be exploited when making a prediction. We do not consider this possibility here.

Putting hierarchical multi-classification into the distance-based context means we need a method for deriving a distance from the hierarchy. Intuitively, the distance between two classes is smaller if they are closer to each other in the hierarchy. Further, the siblings of a node should be equidistant, and the distance from a node to its parent is the same for all the nodes on a given level. Some natural distance definitions fulfilling these criteria will be given below.

The quality criterion $Q$ can but need not be based on the distance. For instance, it could be just the average accuracy with which all the different

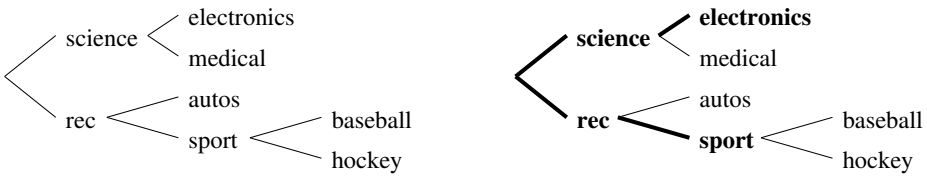


**Fig. 1.** To the left, a class hierarchy; to the right, the prediction $\{rec.sport, science.electronics\}$ is depicted, which forms a subtree of the hierarchy (indicated in bold).

classes are predicted, or it could take into account the fact that predicting `rec.sport.hockey` is a smaller mistake for an article that belongs to class `rec.sport.baseball` than for an article that is not about sport at all.

We finally remark that by representing the prediction as a subtree of the hierarchy, the natural constraints on class membership (anything belonging to A.B.C automatically belongs to A.B and to A) are automatically honoured. This would not be guaranteed if independent models were learnt for all different classes.

Now that the problem is clearly defined, the question is how to construct an algorithm that learns predictive models for this setting. In this paper we follow the predictive clustering approach presented in [4], for which it has been argued that it provides a very general approach to predictive modelling.

## 4 Predictive Clustering Trees

A variety of algorithms for predictive modeling exists. Among the better known are algorithms that induce decision trees [6, 18]. Compared to other well-known techniques such as neural networks [2], decision trees have the advantage of being more interpretable: they clearly explicitate the factors that influence the outcome most strongly.

Decision trees are most often used in the context of classification or single-target regression; i.e., they represent a model in which the value of a single variable is predicted. However, as a decision tree naturally identifies partitions of the data (course-grained at the top of the tree, fine-grained at the bottom), one can also consider a tree as a hierarchy of clusters [12]. A good cluster hierarchy is one in which individuals that are in the same cluster are also similar with respect to a number of observable properties.

This leads to a simple method for building trees that allow the prediction of multiple target attributes at once. If we can define a distance measure on tuples of target variable values, we can build decision trees for multi-target prediction. Similarly, if a distance on structured target values is defined, we can build decision trees for prediction of structural target variables. The methodology has been used successfully for a variety of applications such as conceptual clustering [4], simultaneous prediction of multiple parameters [5], and ranking tasks [23].

The algorithm for inducing such trees is essentially a standard TDIDT (Top-down induction of decision trees) algorithm such as ID3 [17]. The general idea is to recursively partition a set of data into clusters in such a way that the intra-cluster variance is minimized. (In other words, the heuristic for selecting tests to include in a node of the tree is based on this variance.) Intra-cluster variance is defined as the sum of squared distances between the members of the cluster and its prototype $p$, where the latter is defined as $p = \arg\min_q \sum_i d(x_i, q)^2$, i.e. roughly the point that is closest to all the instances in the cluster, according to the distance defined. This prototype may not be a valid prediction. For instance for 0-1 prediction the prototype could be the mean of all target values, e.g. 0.8, but when making a prediction for a specific instance this has to be converted into a valid prediction (0 or 1). The result of the induction process is a decision tree in which each leaf contains (a prediction derived from) the prototype of the examples covered by that leaf.

A detailed description of the algorithm can be found in [4]. The main point to be made here is that the proposed method for inducing predictive clustering trees relies entirely on the definition of the distance measure, the prototypes, and the mapping of prototypes onto valid predictions. These issues are the focus of the next section.

## 5 A distance measure

We introduce a distance measure that is used here for the specific purpose of hierarchical multi-classification, but is in fact more generally useful for distance-based multi-classification. The idea is that from distances between individual classes, which are input to the method, a distance measure between sets of classes is defined that is compatible with these individual distances. This is done by mapping the sets onto vectors in a Euclidean space where the individual classes are base vectors. Depending on the given distances, the base may or may not be orthogonal. (Intuitively, when two classes are close together their base vectors will point more or less in the same direction.)

We first discuss some alternatives for deriving a distance between classes from a hierarchy, next we discuss how it can be upgraded to a distance between sets of classes.

### 5.1 Distance between nodes in a hierarchy

Most of the current work on hierarchical classification considers a distance between 2 nodes of a tree. Examples are:

- shortest path distance (SPD) : considering the tree as a graph, the number of edges on the shortest path between two nodes is the distance between the nodes. It can be computed as $depth(n_1) + depth(n_2) - 2depth(n)$ with $n$ the deepest common ancestor (DCA) of $n_1$ and $n_2$.

- weighted shortest path distance (WSPD) : similar to the previous one, but now edges have weights. Edges deeper in the tree typically have lower weights.
- weighted penalty (WP): weights are assigned to nodes; the distance between two nodes is the weight of their DCA. Nodes deeper in the tree have lower weights.

The difference between WSPD and WP distances is mainly that the first type focuses on dissimilarities between nodes (the parts under their DCA), the second on similarities (the part above their DCA - the larger this is, the smaller the DCA's weight). Note that these distance measures generalise some of those used in the ILP context, e.g. [15].

Which of the above (or other) distances should be used will normally depend on the application; it is not our intention to make general claims about them here. Our method is sufficiently flexible to cater for a wide variety of distance measures, and hence for a wide range of applications.

While we assume that a distance measure for individual nodes in a tree is given, we actually need one for *sets* of nodes. We continue to describe how the distance between nodes is upgraded to a distance between sets of nodes (or rather, "combinations" of nodes, as we shall see).

### 5.2 Distance between combinations of nodes

We assume we have a finite domain of elements $U = \{e_1, \ldots, e_n\}$ (here, the classes). The metric we need is between subsets of $U$. Given a metric between elements of a domain, several metrics between subsets of that domain have been defined, e.g. the Hausdorff metric and the metric proposed in [19] which is based on matchings. However, we will use a decision tree algorithm and hence we would like to be able to compute prototypes efficiently. Since for the existing metrics on sets it is not straightforward to do that, we will choose an approximative method that originates in kernel-based methods.

The empty set is also a subset of $U$ and is called $o$ (the origin). We assume the application allows to define a distance $d_0$ between singletons and the origin ($d_0(e_i, o)$) and between different singletons ($d_0(e_i, e_j)$). Note that these distances respectively correspond to the earlier mentioned $CI_i = CO_i$ and $CS_{ij} = CS_{ji}$.

We follow the procedure proposed in [16]. A set can be converted into a binary vector where the $i$-th component indicates whether element $e_i$ is in the set (1) or not (0). Note that the singletons $\{e_i\}$ are base vectors $\mathbf{e}_i$ in this space (with the $i$-th component 1 and all other components 0).

In this $D$-dimensional feature space a kernel and a corresponding Euclidean distance can be defined that is compatible with all assumed distances $d_0(e_i, o)$ and $d_0(e_i, e_j)$. We can then define the function $k_0 : U \times U \to \mathbb{R}$ by

$$\forall e_i, e_j \in U : k_0(e_i, e_j) = \frac{1}{2}(d_0(e_i, o)^2 + d_0(e_j, o)^2 - d_0(e_i, e_j)^2)$$

$k_0(e_i, e_j)$ is the inner product of $e_i$ and $e_j$ in the Euclidean space.

An element in the feature space $\mathbb{R}^D$ can be written as a linear combination $\Sigma x_i \mathbf{e}_i$ of base vectors, such an element is denoted as $\psi$ and $x_i$ is denoted as $\psi(\mathbf{e}_i)$. Next, we define a function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ by

$$\forall \psi_1, \psi_2 \in \mathbb{R}^D : k(\psi_1, \psi_2) = \sum_{e_i, e_j \in U} \psi_1(\mathbf{e}_i)\psi_2(\mathbf{e}_j)k_0(e_i, e_j).$$

As discussed in [16], if $k$ is positive semi-definite ($\forall \psi \in \mathbb{R}^D : k(\psi, \psi) \geq 0$), one can verify that $k$ is a kernel. Also, the metric $d : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ induced by $k$ and defined by $d(\psi_1, \psi_2) = \sqrt{k(\psi_1 - \psi_2, \psi_1 - \psi_2)}$ agrees with $d_0$, i.e.

$$\forall \mathbf{e}_i, \mathbf{e}_j : d(\mathbf{e}_i, \mathbf{e}_j) = d_0(e_i, e_j)$$

If $k$ is not positive semi-definite, then $d(\psi_1, \psi_2)^2$ is not always non-negative and hence $d$ is not a metric. E.g. consider a set $X = \{a, b, c, d\}$ with $d_0(a, b) = d_0(b, c) = d_0(c, d) = d_0(d, a) = 1$, $d_0(a, c) = 2$ and $d_0(b, d) = 0.1$. If an Euclidean space, then $b$ and $d$ are both in the center of the line connecting $a$ and $c$ (as $d_0(a, c) = d_0(a, b) + d_0(b, c) = d_0(d, a) + d_0(c, d)$). However, $d_0(b, d)$ is different from 0 and it is impossible to map $X$ to an Euclidean space. This is reflected by the fact that the kernel based on this $d_0$ is not positive semi-definite. The positive eigenvalues of the matrix $K$ are on average still larger in absolute value than the negative ones as one starts from a basic distance function $d_0$ which is positive for all pairs of base vectors. As discussed in [16], while such a kernel does not satisfy all required theoretical properties of a metric, it turns out to behave relatively well in practice (it is not well understood why this is so).

In vector notation, we can say that we have constructed a matrix $K$ with elements $K_{ij} = k_0(e_i, e_j)$ such that for any two linear combinations $\psi_{\mathbf{x}} = \Sigma x_i \mathbf{e}_i$ and $\psi_{\mathbf{y}} = \Sigma y_i \mathbf{e}_i$ of elements of $U$ represented as vectors, ${\psi_{\mathbf{x}}}^T K \psi_{\mathbf{y}}$ is their inner product and $d(\psi_{\mathbf{x}}, \psi_{\mathbf{y}}) = (\psi_{\mathbf{x}} - \psi_{\mathbf{y}})^T K (\psi_{\mathbf{x}} - \psi_{\mathbf{y}})$ is the distance between them.

Summarizing, given a distance $d_0$, we can compute a kernel $k$ and thus define a Euclidean distance metric $d$ between sums of class values which is compatible with the given distances in the sense that for all class values $e_i, e_j$ it holds that $d(\mathbf{e}_i, \mathbf{e}_j) = d_0(e_i, e_j)$. For sets $\{e_{i_1}, \ldots, e_{i_k}\}$ and $\{e_{j_1}, \ldots, e_{j_l}\}$ of class values, we can then use $d(\sum_{m=1}^{k} e_{i_m}, \sum_{i=1}^{n} e_{j_n})$ as a distance between these sets. As we will show in the next section, it now becomes easy to define prototypes.

### 5.3 Prototype

In a Euclidean vector space, the prototype of a set of vectors is just the arithmetic mean of the vectors in the set: $\overline{\mathbf{x}} = \sum_{i=1}^{n} \mathbf{x}_i / n$. It is efficient and easy to compute. Note that with non-Euclidean distances, such as Manhattan distances, prototypes are often not uniquely defined or are more difficult to compute.

### 5.4 Mapping a prototype onto a prediction

Suppose we have learned a decision tree that sorts a new example into one of its leaves. We can use this tree to make predictions about the class(es) of these

examples. Therefore, we look at the prototype $\mathbf{p}$ of the training examples in the leaf that is assigned to a new example. As discussed in the previous section, this prototype is a vector in the Euclidean space where the class values were mapped in. This vector is not necessarily a valid prediction, e.g. if we have a set of examples $\{(1,0),(0,1),(0,1)\}$ the mean is $(0.33, 0.67)$ and the components have to be rounded to 0 or 1.

Just rounding all individual components to 0 or 1, whichever is nearest, does not necessarily give an optimal solution because the base vectors may not be orthogonal (i.e. there are dependencies between the components). Therefore we use the following iterative procedure to construct a valid prediction. We start with a valid prediction $P$ that is just the empty set, and in consecutive steps greedily add the base vector that maximally decreases the error criterion $\sum_i d^2(P, e_i) \cdot p_i$ (with $p_i$ the $i$-th component of the prototype) until no more such vectors exist.

## 6 Preliminary Experiments

The experiments we describe here are rather preliminary. We have implemented one version of the distances mentioned above and run experiments with it on two easily obtainable data sets, trying to gain some insight in the behaviour of the proposed method.

### 6.1 Algorithm

For our experiments we use CLUS[1], a system designed for building predictive clustering trees as discussed in Section 4. (It is in fact a propositional version of the Tilde system [3, 4].) We have added a clustering mode to CLUS based on the distance described in Section 5.2. For the distance between base vectors we use a weighted shortest path distance. The weights used in this distance decrease exponentially with the hierarchy depth, i.e. $w_i = w_0^d$, with $w_i$ the weight of a given edge, $d$ the depth of the node from which the edge originates and $w_0$ some parameter. In our experiments we have arbitrarily chosen $w_0 = 0.75$.

### 6.2 Text Classification

**Experimental Setup** In our first experiment we apply CLUS to the task of classifying Usenet newsgroup articles. (We are aware that decision trees are not an optimal tool for document classification, e.g. support vector machines usually perform better; but it is still a reasonable application domain when the aim is to learn something about our approach.)

Articles are always assigned to leaves of the newsgroup hierarchy. Although articles can belong to multiple newsgroups, in the data set we obtained most articles belong to a single newsgroup. Therefore this experiment is useful mainly

[1] CLUS is available from the authors upon request.

| | |
|---|---|
| comp.graphics | rec.autos |
| comp.os.ms-windows.misc | rec.motorcycles |
| comp.sys.ibm.pc.hardware | rec.sport.baseball |
| comp.sys.mac.hardware | rec.sport.hockey |
| comp.windows.x | sci.crypt |
| talk.politics.guns | sci.electronics |
| talk.politics.mideast | sci.med |
| talk.politics.misc | sci.space |
| talk.religion.misc | alt.atheism |
| soc.religion.christian | misc.forsale |

**Table 1.** Usenet newsgroups used in newsgroup data

to investigate the benefit of using hierarchical information when building a multi-classification tree. We compare two settings of CLUS: hierarchical (1) and flat (2) multi-classification. With "flat" classification we denote the setting where all leaf classes are equidistant, i.e., no information about the hierarchy is taken into account.

We use the data set collected by Ken Lang [10]. It contains 20000 articles taken from 20 different newsgroups. In [10] the data set is used to compare different algorithms that can perform flat single classification.

Data preparation involved the following. The original hierarchy for this data set is not ideal in that parts of it degenerate into a flat class space (e.g. the hierarchy contains only one subclass in the category `soc`). In order to increase the hierarchical structure of the class space, we removed the classes `alt.atheism`, `soc.religion.christian` and `misc.forsale`. Further, cross-posted articles appear as multiple examples in the data set, but in our setting they should form a single multi-class example. After removing the above three classes, merging the cross-posted examples, and cleaning up the data set a bit (e.g. removing articles with bodies like "unsubscribe") we end up with a data set containing 16588 examples of which 311 are multi-class.

For each example we selected 1000 features (attributes for CLUS) using the statistical text processing system RAINBOW [13]. The features were selected using the information gain criterion and each feature corresponds to the number of occurrences of a given word in an article. Before passing an article to RAINBOW we removed all header fields except for the subject, because some header fields may reveal the class of the article.

Experiments were then run as follows. We first randomly selected 50% of the available examples as test examples. From the remaining 50% we sampled training sets of different sizes ranging from 100 to 8000 examples. For each training set we compare: (1) "hierarchical": CLUS using the distances as defined by the newsgroup hierarchy; and (2) "flat": CLUS using a distance criterion where all classes are equidistant (this was simulated by redefining the hierarchy so that all classes are on one level, directly below the root.) The results are evaluated by calculating the mean squared weighted shortest path distance $\sum_i d^2(a_i, p_i)$ (MSWSP) between the actual $a_i$ and predicted $p_i$ set of classes over the test

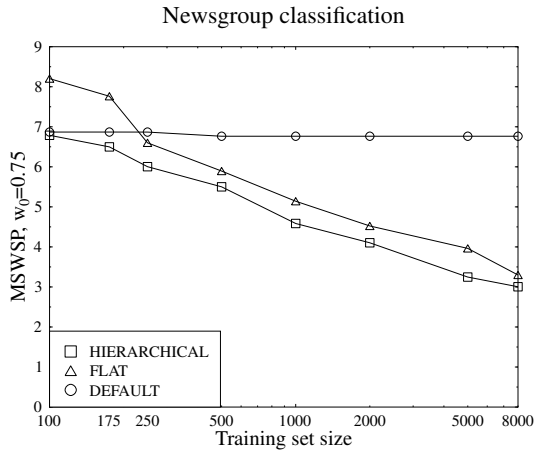


**Fig. 2.** Evolution of shortest path error criterion with training set size. Hierarchical clustering consistently performs slightly better than flat clustering.

examples, according to the true hierarchy. The whole experiment is repeated 10 times and the average results are reported.

**Results** Figure 2 shows the average results obtained. It also shows a line marked with "default" which corresponds to always predicting the set of classes corresponding to the prototype of the training set. (This corresponds to using a decision tree with just one leaf.)

For training sets with a low number of examples, the "flat" method scores worse compared to both "hierarchical" and "default". It scores worse than "default" because it generates sub-optimal predictions, i.e. even if it does not introduce any test, and it would just predict one leaf, then the set of classes predicted in this leaf would differ from "default" because it does not use hierarchical information when calculating prototypes.

If the number of training examples increases then the "hierarchical" approach is always slightly better, compared to the "flat" approach. That it is only slightly better is somewhat disappointing, given that the hierarchical approach uses much more relevant information than the flat approach (in fact its heuristic is directly related to the error criterion, while the flat approach has to use a rough approximation). A possible explanation for the small difference is that the hierarchy for this data set is rather small, so that the "flat" method can find useful tests without depending on any hierarchical information. An obvious extension to this work is of course to retrieve a larger hierarchy from the Usenet newsgroups and repeat the experiment.

Note that we only allow the algorithm to predict leaves of the hierarchy. This means that it can only predict `comp.graphics` and not `comp`. If we allow also

internal nodes, then the MSWSP could be decreased further (for "hierarchical"). We did not do this here because the internal nodes are not valid newsgroups.

### 6.3 Functional Genomics

**Experimental Setup** In our second experiment we apply CLUS to the multi-classification task introduced by Clare and King [8]. They present a modified version of the C4.5 decision tree learner [18] that is capable of learning multi-classification trees but does not exploit any hierarchical information.

The phenotype data set[2] contains 1461 examples. Each example corresponds to a mutant strain that is obtained by removing a specific gene from a cell (in this case yeast *S. cerevisiae* cells). The mutant cells are grown under different conditions or growth media. Each growth medium corresponds to one of the 69 attributes of the data set. Each attribute is three-valued: "wild-type" means that the growth of the mutant is equal to that of the original cells (the wild-type), "sensitive" means less growth for the mutant and "resistance" means better growth for the mutant. The data set has a lot of missing values (84.41% of all attribute values is missing) because not all possible growth experiments have been carried out. Each example has also an attribute "notn" which is the discretized number of growth media where the mutant differs from the wild type. The target value is the set of functional classes of the removed gene. This functional class belongs to a hierarchy of depth 4 with 13 classes at level one (e.g. "metabolism", "energy", "transcription", ...) and 162 leaves.

In [8] a resampling approach is used to find accurate and stable rules, and some selected rules are presented. In our preliminary experiment we use just one train/test split (50% of the examples each). We use CLUS with the hierarchical multi-classification setting to build a model on the training set and see how this compares with some of the rules presented in [8].

**Results** Figure 3 shows a part of the clustering tree we obtained. The tree is very small: it contains only two tests. Apparently few tests are considered significant, probably because of the huge number of missing values in the data.

CLUS shows for each test (i.e. lines 1 and 2 of Figure 3 ) the percentage of examples for which the test succeeds and the percentage of missing values for the attribute in the test. For each leaf it shows (after "size") the expected number of training examples and test examples in the leaf[3], and the number of test examples that *certainly* belong to the leaf (i.e. that have no missing values for the attributes used in the tests above the leaf). Furthermore for each leaf it outputs a list of classes with their expected frequency among test examples possibly in the leaf, test examples certainly in the leaf, and all test examples.

For instance, the first leaf covers examples for which `calcofluor_white = resistance` and `notn = 1`. The expected number of test examples in this leaf

[2] Available at `http://users.aber.ac.uk/ajc99/phenotype/`

[3] These can be real numbers because examples with missing values belong only with a certain probability to the leaf.

```
1  calcofluor_white = resistance (2.3%) (miss: 26.9%)
2  +--yes: notn = 1 (57.3%) (miss: 0%)
3  |       +--yes: size: 7.8, 5.1, 2
4  |       |       [1: 0.5898, 1, 0.3292]
5  |       |       [1/5: 0.4485, 1, 0.1311]
6  |       |       [1/5/1: 0.2448, 0.5, 0.0915]
7  |       |       [1/5/4: 0.1992, 0.5, 0.0383]
8  |       |       ...
9  |       |
10 |       +--no: size: 5.8, 6, 5
11 |              ...
12 |              [9: 0.3355, 0.4, 0.0779]
13 |              [9/1: 0.3317, 0.4, 0.0601]
14 |              ...
15 |
16 +--no: size: 570.3, 718.8, 545
17        [1: 0.325, 0.3211, 0.3292]
18        ....
```

**Fig. 3.** A clustering tree grown on the phenotype data.

is 5.1, and there are 2 test examples for which we are certain they belong in this leaf. The information on class 1/5 (*C-compound and carbohydrate metabolism*) indicates that 44.85% of the examples sorted in this leaf belong to class 1/5. Of the two examples that certainly belong in this leaf, both (100%) belong to class 1/5. In the whole test set only 13.11% belongs to class 1/5, so for two random examples in the test set there would be about a 1.7% chance that both are in class 1/5. Thus the rule "if calcofluor_white=resistance and notn=1 then the gene has function 1/5" is validated with high significance. Even if no examples were guaranteed to belong to the leaf, we could still see that among all the examples that may or may not belong to the leaf (weight 5.1) there is an increased percentage that belongs to class 1/5 (44.85% compared to 13.11%).

The accuracy and confidence of the above rule is comparable to that of the rules presented in [8]. They present e.g. the following rule: "If the gene is resistant to calcofluor white then its class is 9/1: biogenesis of cell wall (cell envelope)". The single condition in this rule corresponds to the condition in the top node of our tree. The rule covers on average 6.7 examples with their resampling strategy, and 43.8% of these examples have class 9/1 (compared to 9.5% in the whole data set) [8].

Note that by predicting classes at one single level of the hierarchy, it is difficult to see whether the condition is really related to class 9/1, or rather to its superclass 9 or one of its subclasses $9/1/x$. Our tree, on the other hand, gives information for classes on different levels of the hierarchy, e.g. 1, 1/5, 1/5/1. Statistics for a class can be compared to those of its subclasses to see at which level the deviations truly occur. For instance, looking at the first leaf in Figure 3: the deviation for class 1 (all of the certainly covered examples belong

to it, instead of 33%) can entirely be attributed to class 1/5 (all certainly covered examples belong to 1/5), but the deviation for 1/5 cannot be attributed to a single of its subclasses (1/5/1 and 1/5/4 both cover one example). In our current implementation it is left to the user to perform such comparisons, but it could easily be automated.

## 7 Related work

Several authors have generalised decision trees to the multi-classification setting, see e.g. Suzuki et al.'s bloomy decision trees [22] and the functional genomics application by Clare and King who present a multi-classification upgrade of C4.5 [8]. In both cases the notion of class entropy is extended to cover multiclassification. The main difference with our approach is that our clustering approach naturally allows us to take into account hierarchical information.

Hierarchical approaches in the area of text classification include [24] (a single-class approach based on association rules) and [11] (which builds a separate predictive model for each node). We are not aware of existing approaches that learn a single model in a context that is both hierarchical and multi-class.

Prediction of structural information is to some extent related to case-based reasoning, which can be seen as an extension of instance-based learning to the structural prediction setting (see e.g. [14]). Other work on structure prediction includes Ramon and De Raedt's "instance based function learning" [21].

The distance measure we propose is not the only possible one; as mentioned before many distance measures for structural values have been proposed, see e.g. [15, 9, 20].

## 8 Conclusions

We have discussed the task of hierarchical multi-classification, which extends both multi-classification and hierarchical classification, and which we believe to cover an interesting range of applications. We have presented an algorithm that extends the decision tree approach towards hierarchical multi-classification. The main assumption in this approach is that from a hierarchy, a natural distance between elements of the hierarchy can be defined. Thus, the approach is a special case of a more general distance-based approach to multi-classification.

We have experimentally validated our approach on two different data sets, one in the area of document classification and one in the area of functional genomics. Both data sets turn out to be somewhat limited with respect to the information they may give us: the first one has a very simple hierarchy and contains few instances with multiple classes; the other has relatively few examples with much missing values, compared to the complexity of the hierarchy. The experiments give some indication as to the usefulness of our approach (especially on the functional genomics data set our interpretation of the results suggests this is a promising approach), but it is clear that more experiments are needed.

Further work will obviously include a more thorough evaluation of the approach on a variety of test sets, as well as an evaluation of which distances work well in which application domains (the ones we used here were relatively ad hoc). We expect to continue to focus our attention on decision tree induction, but our approach could also be extended to e.g. rule based learners. An obvious further direction of research is incorporating the proposed kernel in a support vector machine, and see to what extent the performance of SVM's for text classification could improve through the use of hierarchical information.

Besides hierarchical multi-classification, this paper also points to a possible direction of research regarding the more general problem of prediction of sets of values when a distance measure between individual values is defined, or more generally a cost for certain types of errors.

**Acknowledgements**

Jan Struyf is a research assistant, Hendrik Blockeel a post-doctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). Jan Ramon is supported by the Flemish Institute for the Advancement of Scientific and Technological Research in Industry (IWT). This research is supported by the Research Fund K.U.Leuven.

## References

1. B. Bakker and T. Heskes. Task clustering for learning to learn. In B. Kröse, M. De Rijke, G. Schreiber, and M. Van Someren, editors, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence*, pages 33–40, Amsterdam, 2001.
2. C. M. Bishop. *Neural Networks for Pattern Recognition*. University Press, Oxford, 1999.
3. H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
4. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, 1998. `http://www.cs.kuleuven.ac.be/~ml/PS/ML98-56.ps`.
5. H. Blockeel, S. Džeroski, and J. Grbović. Simultaneous prediction of multiple chemical parameters of river water quality with tilde. In *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, volume 1704 of *Lecture Notes in Artificial Intelligence*, pages 32–40. Springer, 1999.
6. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.
7. R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
8. A. Clare and R.D. King. Knowledge discovery in multi-label phenotype data. In L. De Raedt and A. Siebes, editors, *5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2001)*, volume 2168 of *Lecture Notes in Artificial Intelligence*, pages 42–53. Springer-Verlag, 2001.
9. A. Hutchinson. Metrics on terms and clauses. In *Proceedings of the 9th European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, pages 138–145. Springer-Verlag, 1997.

10. T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-1997 - The 14th international conference on M achine Learning*, 1997.
11. D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, TN, USA, 1997. Morgan Kaufmann.
12. P. Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996.
13. Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering.
`http://www.cs.cmu.edu/~mccallum/bow`, 1996.
14. T. Mitchell. *Machine Learning*. McGraw-Hill, 1996.
15. S.-H. Nienhuys-Cheng. Distance between Herbrand interpretations: A measure for approximations to a target concept. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1997.
16. Elzbieta Pekalska, Pavel Paclik, and Robert P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Artificial Intelligence Research*, 2:175–211, December 2001.
17. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
18. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in machine learning. Morgan Kaufmann, 1993.
19. J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37:765–780, 2001.
20. J. Ramon, M. Bruynooghe, and W. Van Laer. Distance measures between atoms. In *Proceedings of the CompulogNet Area Meeting on 'Computational Logic and Machine Learning'*, pages 35–41, 1998.
21. J. Ramon and L. De Raedt. Instance based function learning. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, pages 268–279. Springer-Verlag, 1999.
22. E. Suzuki, M. Gotoh, and Y. Choki. Bloomy decision trees for multi-objective classification. In L. De Raedt and A. Siebes, editors, *Principles of Data Mining and Knowledge Discovery, Proceedings of the 5th European Conference*, volume 2160 of *Lecture Notes in Artificial Intelligence*, pages 436–447. Springer-Verlag, 2001.
23. L. Todorovski, H. Blockeel, and S. Džeroski. Ranking with predictive clustering trees. In *Proceedings of the 13th European Conference on Machine Learning*. Springer-Verlag, 2002.
24. Ke Wang, Senqiang Zhou, and Shiang Chen Liew. Building hierarchical classifiers using class proximity. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and Brodie M. L., editors, *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 363–374. Morgan Kaufmann, 1999.